Assignment 2

Team Number: 43

Name	Student number	Email
Sanam Izadi	2729395	s.izadi@student.vu.nl
Tanav Rawal	2725239	t.rawal@student.vu.nl
Filip Bickelhaupt	2713594	f.m.bickelhaupt@vu.nl
Denay Araya	2717149	d.araya@student.vu.nl

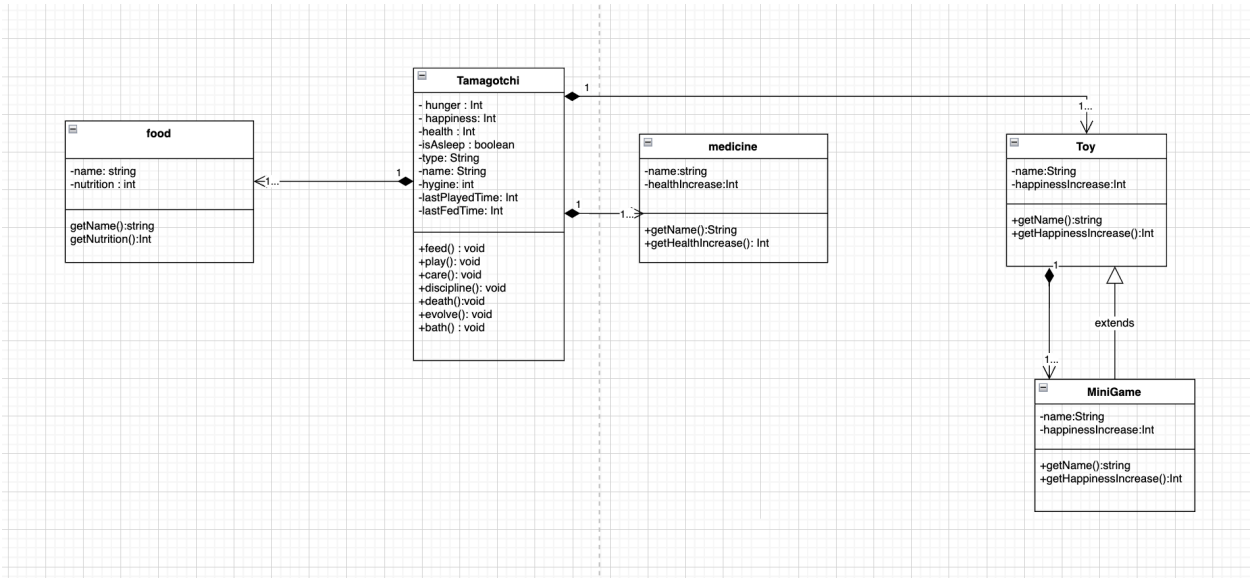
Summary of changes from Assignment 1

Author(s): Tanav Rawal

- We added a more in-depth explanation to the overview part of the game and added that the game is about the user taking care of a pet through different commands.
- We added the functionality of death of the Tamagotchi as it is a must for this project but also because it is an important function of the game in itself.
- We also changed the QR3 quality requirement's quality attribute to scalability as it was more appropriate to the Quality requirement we were trying to describe.

Class Diagram:

Author(s) : Sanam Izadi, Tanav Rawal



We have a total of 4 classes in this class diagram for our game of “MyTamagotchi”:

Tamagothchi class description - The tamagotchi class is the class for the actual virtual pet in the game, which gives the name of the pet, its current levels of hunger, happiness, health, and if it is asleep or not. Apart from these basic descriptive properties about the virtual pet we also have certain actions in the class that the tamagotchi can perform, which affect the statistics of the pet in certain ways, good or bad. Some of the relevant attributes which are going to be used in this class is feed(), play(), care(), discipline, death(), evolve(), and bath(). These are the primary functions in the class that we are going to apply. Feed() of course reduces the hunger stat of the pet, but also increases the happiness stat, and with play() the happiness levels or increase but at a certain point where we can see that the tamagotchi is misbehaving, which will be indicated through the statistics, then the use of discipline() functions comes into play, where it reduces happiness giving you more of a sense of control of the pet. Death() function is quite self explanatory which comes into play automatically when certain stats are not fulfilled which would cause the death of the pet. Evolve() function is when the pet reaches certain stats, the player is awarded a evolution of their pet which shows that their pet has become a doctor, engineer, etc. In terms of the associations in the class, it is mostly of the composition type as most of the other classes are dependent on the tamagotchi class.

Food class description - The food class in the game provides a way for the pet to eat and interact with the different types of food that we have, the particular food item basically shows the nutrition facts and what kind of stats it will increase and or decrease upon consumption from the pet. In order for the food class to work it is associated with the tamagotchi class, for the interactivity with the pet and to change the stats of the pet accordingly. The attributes in the class are the name of the food item, the nutrition facts of the food. The methods which will be

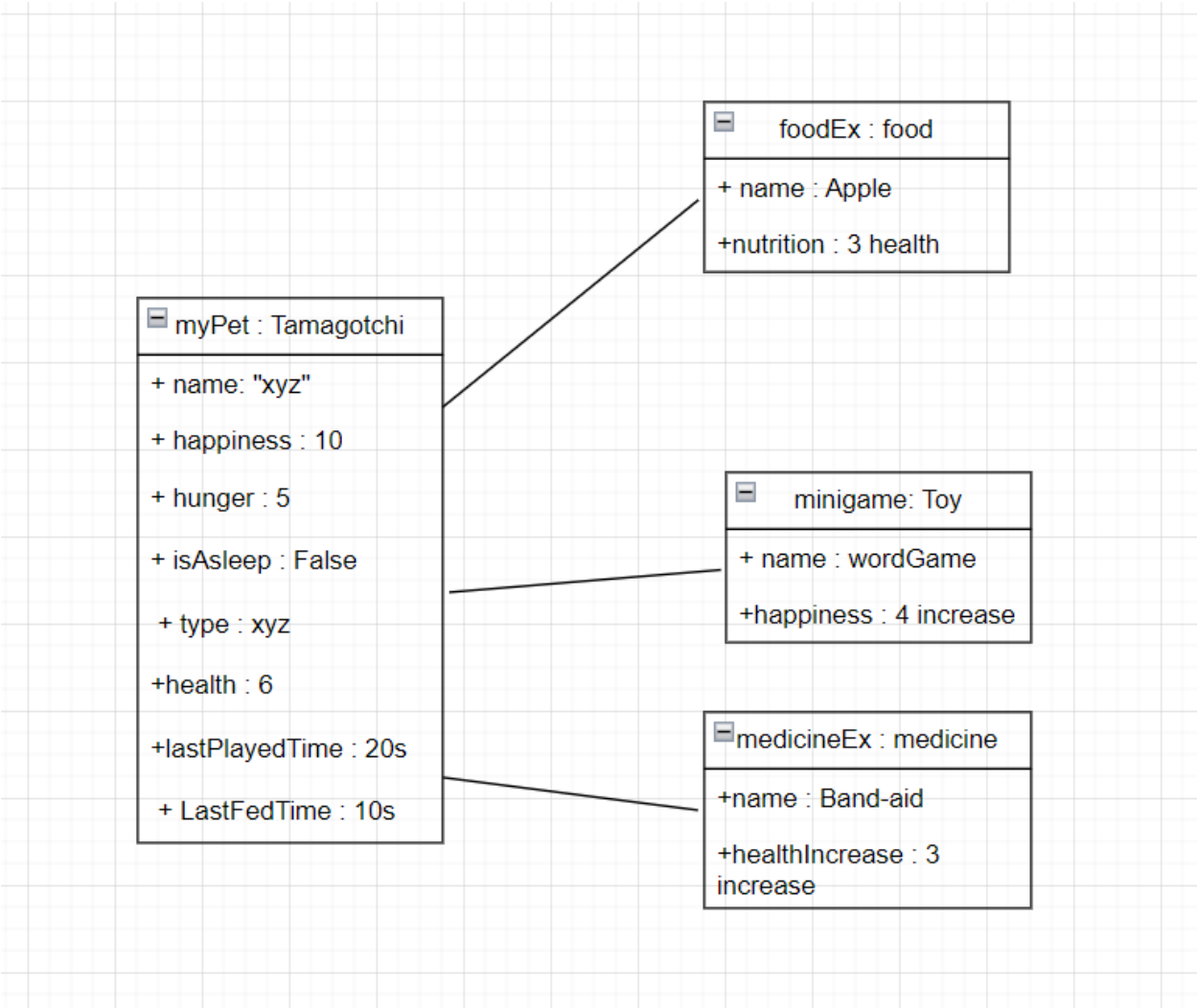
used often with this class are getName(), which by the name explains that it will get the name of the food item from the other food items in the list of foods. Along with this there is getNutrition() which shows the nutrition fact of the particular food item. This is of course associated with the tamagotchi class using the concept of composition again from the previous class.

Medicine class description - The medicine class basically represents in the game in a similar way the food class does. It basically specifies the different type of medicine items which can be used to boost certain stats for the pet. Now in order for this class to effect the tamagotchi class, it is associated with the tamagotchi class in the same way the food class is as well. All in all the medicine class provides a very important part of the game, by providing medical items when the pet needs it, it could also be used by the user to prevent death of the pet in the game. The attributes that this class offers are name and healthIncrease, which by the names are pretty self explanatory and provides the names for the particular medical item and the amount of the healthIncrease it is able to provide. Now of course with these attributes there are certain methods along with it as well. One is getName() similar to food class, gets the name of the medicine item. And getHealthIncrease() which gets the information of what stats need to be changed in the pet.

Toy class description - The toy class represents the way the pet can play with a toy within the actual tamagotchi game. Within the toy class, there are different types of toy items that will be prompted as different options for the player to choose from for their personal pet to interact with. In order for the Tamagotchi to interact with the toy which is being provided, there needs to be an association which is again going to be composition as 1 to many relationships as there is only one tamagotchi class but many different types of toys for the player. There are some basic attributes which are included with this class as well, such as the name of course which provides the name of the toy which will be given to the user to choose from different other toys. Along with this the toy brings a certain amount of happiness points, which basically means it increases the happiness meter of the pet. It is the same for the methods which are being used which are getName and getHappinessIncrease() which simply gets the information which is necessary for the item which is being searched by the user. Minigame is more of a subclass as it is also like a side game which the pet can play, it can also be considered as a toy however it has its own attributes which makes it to vague for it to be just an object of the toy class.

Object Diagram:

Authos(s): Sanam Izadi, Tanav Rawal



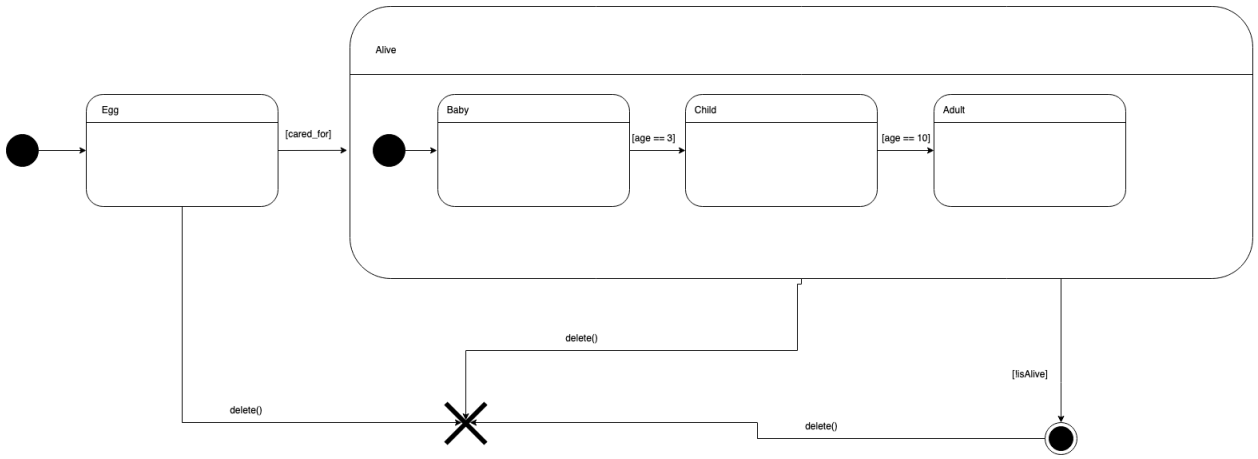
In the object diagram above we are able to describe a snapshot of the state of the game when we show the connection of the other classes which depend on the main tamagotchi class. We are using links for this purpose to show the relationship between all the classes from the previous class diagram. From this we are able to see an example state of the game. With a sample name of xyz, we can see the certain action with the other classes from the associations that we have set up in the class diagram. In the minigame we are able to see that it is an instance of the toy class, as it is a subclass, and it has basic attributes such as name and the amount of stats it increases. Similarly with the medicine and food objects that we have created as an example for this diagram, we have an “apple” food which gives a total of 3 nutrition points in the health stat of the pet in the final and in the medicine there is an health increase of 3 if the pet chooses to

interact with it. With this we are able to show a small part of the how the classes work together with a very basic example of how it would work.

State machine Diagram:

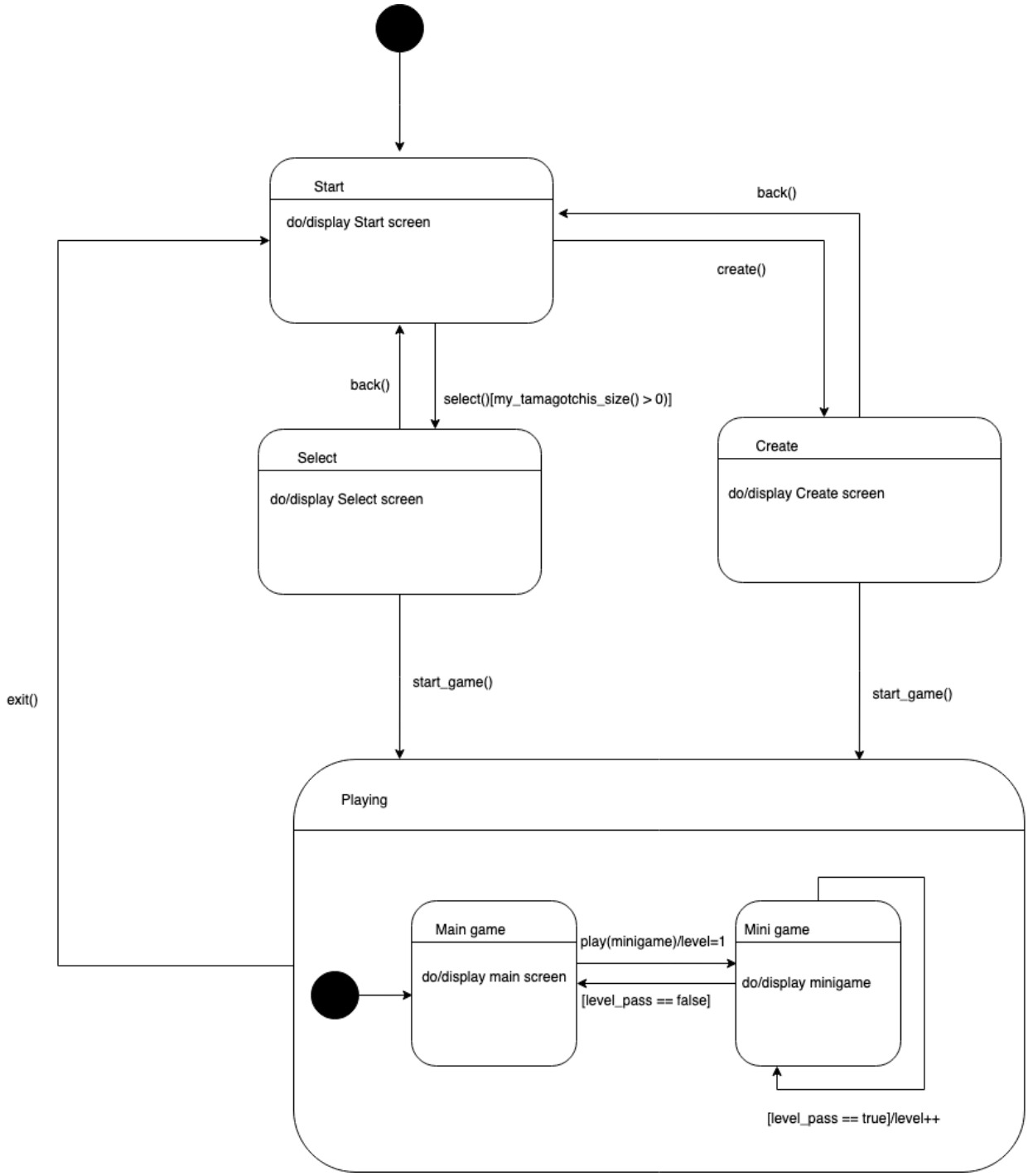
Author(s) : Filip Bickelhaupt

Diagram 1, state of tamagotchi



This diagram describes the lifespan of a single tamagotchi after it has been created. It starts as an egg and in this state the player can care for the egg by inputting a certain key, for example 5 times to keep it simple. Then it will be a baby from this state forward the player will have all actions available. The tamagotchi will go to the next state by reaching a certain age. When it dies it reaches a final state. The tamagotchi can get deleted in any state.

Diagram 2, state of the game

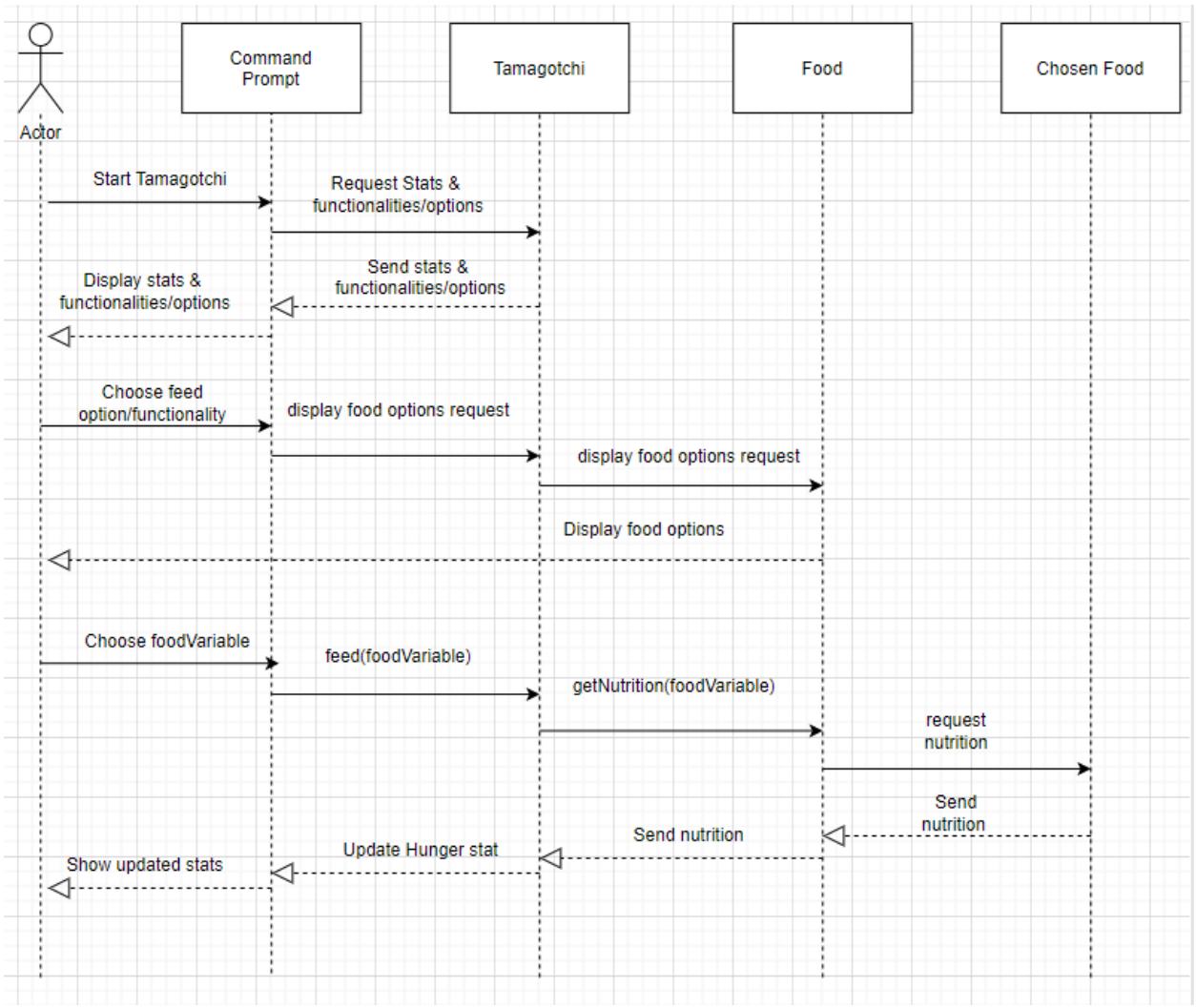


This diagram models the state that the player is in within the game. The player starts in the start state, here the player is shown a welcoming message and is explained that he has the option to either select an already created tamagotchi or create a new one. The player can only choose to select a tamagotchi if they have already created at least one tamagotchi. In the select state the player is shown a list of numbers and their tamagotchi and choose which one they want to play as by inputting the tamagotchi's corresponding number. In the create state the player can create their new tamagotchi. After the select and create classes the player enters the playing state

where they start in the main game state, here they are displayed a UI with all the commands he needs to play the game. If the player inputs a command for the minigame, a minigame will start. In the minigame state the player will keep reaching higher levels while they pass the levels and go back to the main game state when they lose.

Sequence diagram:
Author(s): Denay

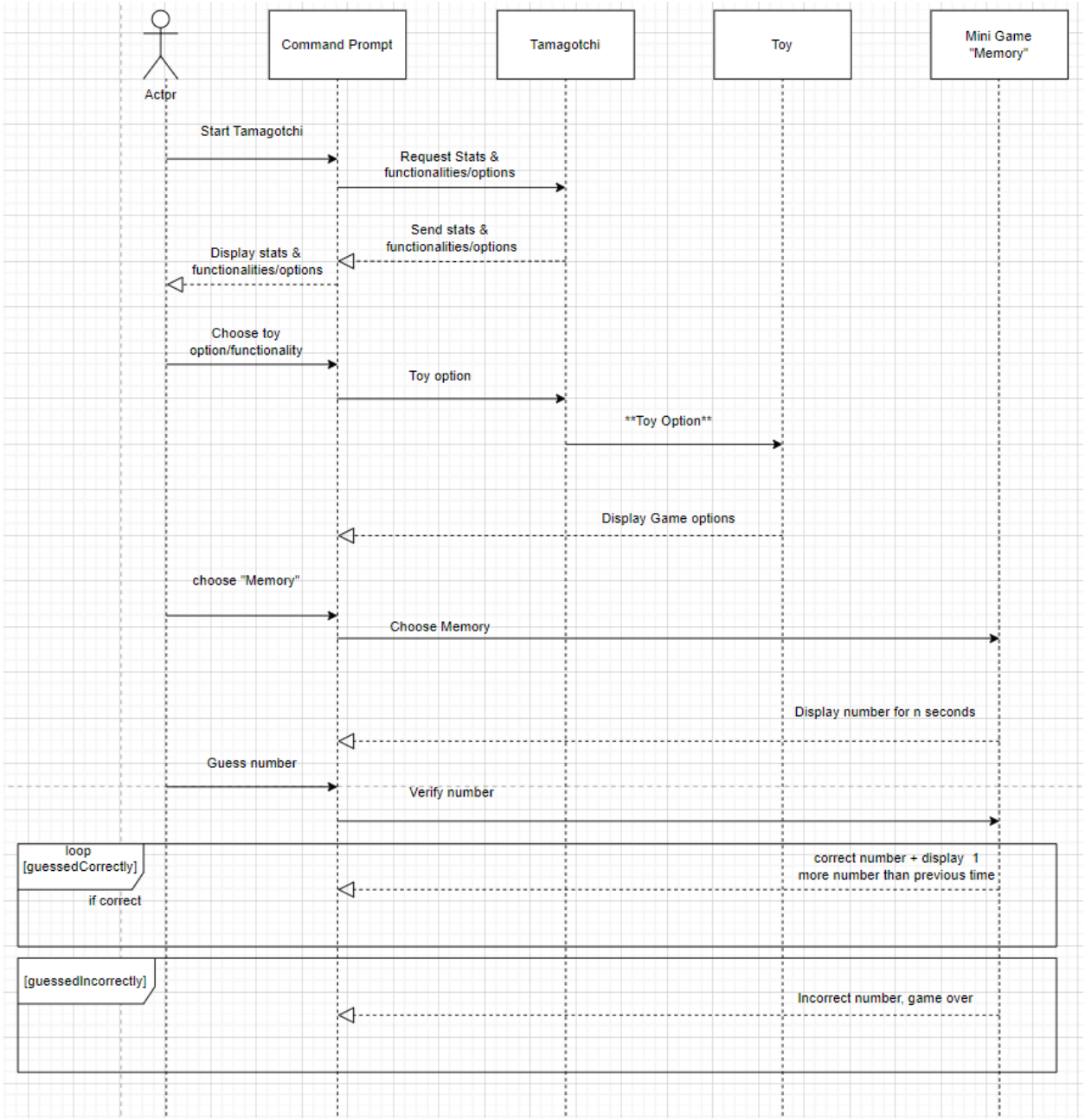
Diagram 1 (Feeding)



The above sequence diagram displays the order of interactions between the actor, command prompt and the objects, when an actor wants to feed the tamagotchi. First the actor starts up the whole system by typing a not yet decided command into the command prompt(this could be start) then from the tamagotchi class all options will be displayed on the command prompt. Next to every option it says the command that needs to be typed in the command prompt to choose the given option(This form of interaction may change). The actor will then choose the feeding option which via the tamagotchi class will get him in the food class. The food class will also display several options. These options are chicken sandwich, apple and banana. Each with their

own nutritional value represented by an integer. After the food is chosen the tamagotchi class will perform the feed function with the chosen food. For his function the tamagotchi class needs to know the nutrition integer which it requests from the food class. The food class sends back the nutrition value from the chosen food object which the feed function in the tamagotchi class uses to update the hunger stat. Then the feed function displays the updated states, showing that the tamagotchi is less hungry.

Diagram 2 (Playing)



(The above diagram ,and thus the following description, are very similar to the previous sequence diagram up until the “Choose toy option/functionality” part) The above sequence diagram displays the order of interactions between the actor, command prompt and the objects, when an actor wants to feed the Tamagotchi. First the actor starts up the whole system by typing a not yet decided command into the command prompt(this could be start) then from the tamagotchi class all options will be displayed on the command prompt. The actor will then choose the toy option. The Tamagotchi will then access the toy class which will display all the mini game options on the command prompt. The actor will then choose the memory game which the mini game class will then start. The minigame class will display a number for a certain amount of time afterwards the actor will type the given number in the command prompt which

the mini-game class will then verify. If the number is correct there will be an extra number displayed which the user needs to remember and type out(so instead of 1 number 2 numbers will be displayed). as long as the user keeps correctly guessing these instructions will keep on looping. When the number is wrong the minigame class will print that the number is wrong and end the game